AE 04: NYC flights + data wrangling

David Bell

```
library(tidyverse)
library(nycflights13)
```

Exercise 1

Your turn: Fill in the blanks:

The `flights` data frame has **336,776 rows**. Each row represent a different flight departing from New York City.

Exercise 2

Your turn: What are the names of the variables in `flights`.

```
colnames(flights)
```

```
 [1] "year"           "month"          "day"            "dep_time"
 [5] "sched_dep_time" "dep_delay"      "arr_time"       "sched_arr_time"
 [9] "arr_delay"      "carrier"        "flight"         "tailnum"
[13] "origin"         "dest"           "air_time"       "distance"
[17] "hour"           "minute"         "time_hour"
```

Exercise 3 - `select()`

- Make a data frame that only contains the variables `dep_delay` and `arr_delay`.

```
deparr_delay <- select(flights, c(dep_delay, arr_delay))
```

- Make a data frame that keeps every variable except `dep_delay`.

```
flight_3b <- select(flights, c(1:5,7:19))
```

- Make a data frame that includes all variables between `year` through `dep_delay` (inclusive). These are all variables that provide information about the departure of each flight.

```
flight_3c <- select(flights, c(1:6))
```

- Use the `select` helper `contains()` to make a data frame that includes the variables associated with the arrival, i.e., contains the string `"arr\_"` in the name.

```
flights_3d <- flights %>% select(contains("arr"))
```

Exercise 4 - `slice()`

- Display the first five rows of the `flights` data frame.

```
f4a <- slice(flights, 1:5)
```

- Display the last two rows of the `flights` data frame.

```
tail(flights, 2)
```

```
# A tibble: 2 x 19
   year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
  <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
1  2013     9    30       NA           1159        NA       NA           1344
2  2013     9    30       NA            840        NA       NA           1020
# i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
#   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
#   hour <dbl>, minute <dbl>, time_hour <dttm>
```

Exercise 5 - `arrange()`

- Let's arrange the data by departure delay, so the flights with the shortest departure delays will be at the top of the data frame.

```
flights_5a <- arrange(flights, dep_delay)
```

- Question: What does it mean for the `dep_delay` to have a negative value?

Add your response here.

- Arrange the data by descending departure delay, so the flights with the longest departure delays will be at the top.

```
flights_5b <- arrange(flights, desc(dep_delay))
```

- **Your turn:** Create a data frame that only includes the plane tail number (`tailnum`), carrier (`carrier`), and departure delay for the flight with the longest departure delay. What is the plane tail number (`tailnum`) for this flight?

```
flights_5c <- flights %>%
  mutate(across(c(tailnum, carrier))) %>%
  arrange(desc(dep_delay)) %>%
  select(tailnum, carrier)
```

Exercise 6 - `filter()`

- Filter for all rows where the destination airport is RDU.

```
flights_6a <- filter(flights, dest == "RDU")
```

- Filter for all rows where the destination airport is RDU and the arrival delay is less than 0.

```
flights_6b <- flights %>%
  filter(dest == "RDU" | arr_delay < "0")
```

- **Your turn:** Describe what the code is doing in words.

The code below is used to filter for destinations going to RDU and GSO in the flight dataset, while not creating a new datafame. The code is also filtering any delays to arrivals and departures to be less then 0.

```
flights |>  filter(dest %in% c("RDU", "GSO"), arr_delay < 0 | dep_delay < 0)
```

```
# A tibble: 6,203 x 19
    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
 1  2013     1     1      800            810       -10      949            955
 2  2013     1     1      832            840        -8     1006           1030
 3  2013     1     1      851            851         0     1032           1036
 4  2013     1     1      917            920        -3     1052           1108
 5  2013     1     1     1024           1030        -6     1204           1215
 6  2013     1     1     1127           1129        -2     1303           1309
 7  2013     1     1     1157           1205        -8     1342           1345
 8  2013     1     1     1317           1325        -8     1454           1505
 9  2013     1     1     1449           1450        -1     1651           1640
10  2013     1     1     1505           1510        -5     1654           1655
# i 6,193 more rows
# i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
#   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
#   hour <dbl>, minute <dbl>, time_hour <dttm>
```

Hint: Logical operators in R:

operator	definition
`<`	is less than?
`<=`	is less than or equal to?
`>`	is greater than?
`>=`	is greater than or equal to?
`==`	is exactly equal to?
`!=`	is not equal to?
`x & y`	is x AND y?
`x | y`	is x OR y?
`is.na(x)`	is x NA?
`!is.na(x)`	is x not NA?
`x %in% y`	is x in y?
`!(x %in% y)`	is x not in y?
`!x`	is not x? (only makes sense if `x` is `TRUE` or `FALSE`)

Exercise 7 - `count()`

- Create a frequency table of the destination locations for flights from New York.

```
flights_7a <-filter(flights, origin == "JFK" | origin == "LGA")
flights_7a
```

```
# A tibble: 215,941 x 19
    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
 1  2013     1     1      533            529         4      850            830
 2  2013     1     1      542            540         2      923            850
 3  2013     1     1      544            545        -1     1004           1022
 4  2013     1     1      554            600        -6      812            837
 5  2013     1     1      557            600        -3      709            723
 6  2013     1     1      557            600        -3      838            846
 7  2013     1     1      558            600        -2      753            745
 8  2013     1     1      558            600        -2      849            851
 9  2013     1     1      558            600        -2      853            856
10  2013     1     1      558            600        -2      924            917
# i 215,931 more rows
# i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
#   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
#   hour <dbl>, minute <dbl>, time_hour <dttm>
```

```
freqtable7a <- count(flights_7a, flights_7a$dest)
print(freqtable7a)
```

```
# A tibble: 94 x 2
   `flights_7a$dest`     n
   <chr>             <int>
 1 ABQ                 254
 2 ACK                 265
 3 ATL               12193
 4 AUS                1471
 5 AVL                  10
 6 BGR                 375
 7 BHM                 297
 8 BNA                3997
 9 BOS               10181
10 BQN                 599
# i 84 more rows
```

- In which month was there the fewest number of flights? How many flights were there in that month?

```
freqtable7b <- count(flights,flights$month)
print(freqtable7b)
```

```
# A tibble: 12 x 2
   `flights$month`     n
             <int> <int>
 1               1 27004
 2               2 24951
 3               3 28834
 4               4 28330
 5               5 28796
 6               6 28243
 7               7 29425
 8               8 29327
 9               9 27574
10              10 28889
11              11 27268
12              12 28135
```

- **Your turn:** On which date (month + day) was there the largest number of flights? How many flights were there on that day?
- November 27th had the most flights departing with 1,014

```
freqtable7c <- count(flights,flights$month, flights$day)
print(freqtable7c)
```

```
# A tibble: 365 x 3
   `flights$month` `flights$day`     n
             <int>         <int> <int>
 1               1             1   842
 2               1             2   943
 3               1             3   914
 4               1             4   915
 5               1             5   720
 6               1             6   832
 7               1             7   933
 8               1             8   899
 9               1             9   902
10               1            10   932
# i 355 more rows
```

```
max(freqtable7c)
```

```
[1] 1014
```

```
view(freqtable7c)
```

Exercise 8 - `mutate()`

- Convert `air_time` (minutes in the air) to hours and then create a new variable, `mph`, the miles per hour of the flight.

```
flight8a <- flights %>%
  mutate(air_time = air_time/60)
mph <- c(flight8a$distance/flight8a$air_time)

flight8a$mph <- flight8a
```

- **Your turn:** First, count the number of flights each month, and then calculate the proportion of flights in each month. What proportion of flights take place in July?
- Create a new variable, `rdu_bound`, which indicates whether the flight is to RDU or not. Then, for each departure airport (`origin`), calculate what proportion of flights originating from that airport are to RDU.

```
flight8c <- flights
flight8c$rdu_bound <- flight8c
flight8c <- mutate(flight8c, rdu_bound = if_else(dest == "RDU", "Yes", "No"))
```

Exercise 9 - `summarize()`

- Find mean arrival delay for all flights.

```
# add code here
```

Exercise 10 - `group_by()`

- Find mean arrival delay for for each month.

```
# add code here
```

- **Your turn:** What is the median departure delay for each airports around NYC (`origin`)? Which airport has the shortest median departure delay?

```
# add code here
```

Additional Practice

Try these on your own, either in class if you finish early, or after class.

1. Create a new dataset that only contains flights that do not have a missing departure time. Include the columns `year`, `month`, `day`, `dep_time`, `dep_delay`, and `dep_delay_hours` (the departure delay in hours). *Hint: Note you may need to use `mutate()` to make one or more of these variables.*

```
# add code here
```

2. For each airplane (uniquely identified by `tailnum`), use a `group_by()` paired with `summarize()` to find the sample size, mean, and standard deviation of flight distances. Then include only the top 5 and bottom 5 airplanes in terms of mean distance traveled per flight in the final data frame.

```
# add code here
```